Projects Problems

1. Combinatorial Problem (Hidato)
2. Binpacking & Scheduling/Planning Problems (Trucks loading & Product Planning)

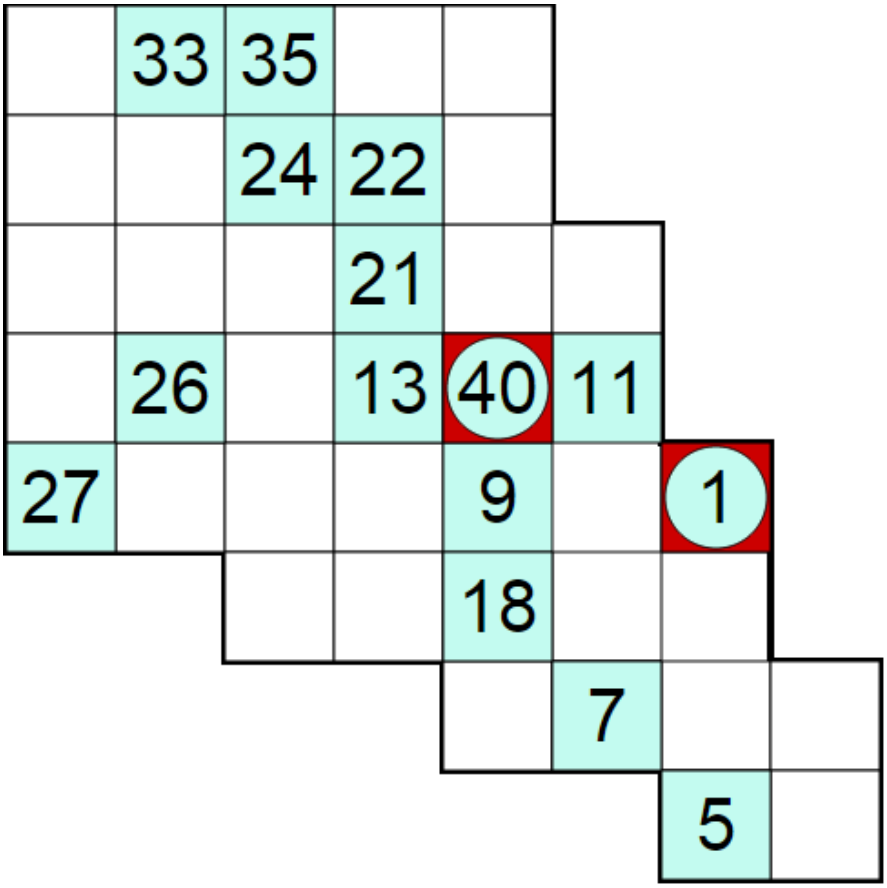
Hidato Puzzle

It is a kind of logic puzzle game and also known as “Hidoku”. The goal of Hidato is to fill the grid with consecutive numbers that connect horizontally, vertically, or diagonally.

In Hidato, a grid of cells is given. It is usually square-shaped, like Sudoku or Kakuro, but it can also irregular shaped grids and it has to be made of only one piece.

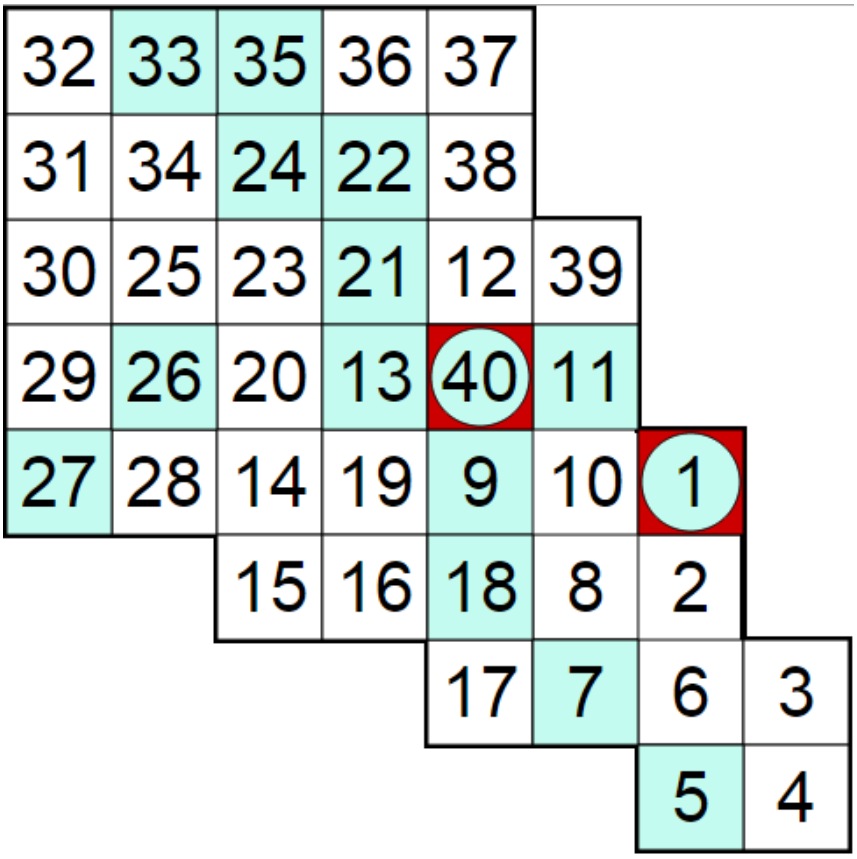
In every Hidato puzzle the smallest and the highest numbers are given on the grid. There are also other given numbers on the grid to help direct the player how to start the solution and to ensure that Hidato has a single solution.

As an example problem, we choose this picture:



For this example 1 is minimum number, 40 is maximum number and others given to help user to find direction.

Solution of this puzzle is in this picture.



Modelling Strategy of Hidato Puzzle

First we define an integer array with given numbers and it represents our grid. Next, we define generic ArrayList for IntVar variables and an IntVar array. After definition of variables, we start to analyse grid. According to given minimum and maximum numbers, domains are implemented for IntVar variables (from 1 to 40).

Then we check the whole grid cell by cell.

- If cell has a given number, program groups the cells three by three and impose ‘Among’ constraint. It means that grouped cells have to contain consecutive numbers.
- If cell is empty and it can be usable, program impose needed constraints which build primitive constraints. These constraints says that if “V” is chosen box value, one of the boxes which is near of current box, has to contain “V+1” and “V-1” value.

Both situations are similar but we need constant value for using “Among” constraint. Because of that, we create similar constraint with using primitive constraints.

At the end of this problem, we make search in order to find solution. At the search method we use some special methods taken from JaCoP library. SmallestMin, SmallestDomain, and IndomainMin are some of them.

(Our codes are in HidatoSolver.java file)

Binpacking & Scheduling/Planning Problems

In the **bin packing problem**, objects of different volumes must be packed into a finite number of bins or containers each of volume *V* in a way that minimizes the number of bins used.

In our example problem, we assume that there are 3 resources different and undivisible sizes. We have also 10 trucks with 10 units capacity and 5 trucks with 15 units capacity. First part of our problem is how we can loads resources to trucks.

In the second part, we have 3 products which is produced from that 3 resources and each product has constant income. Second part of our problem is how we can schedule the product plan for maximum income.

Modelling Strategy of Binpacking Problem

First we define needed variables like ‘Trucks’, ‘Resources’ etc. Then in the first model method, we implement Trucks and Resources IntVar arrays and impose binpacking constraint. Binpacking constraint takes 3 arguments (Truck Numbers,Trucks, integer array of items). Then we found solution with search method. *(Important methods are model and search methods which are located in BinPackProduct.java file)*

Modelling Strategy of Scheduling/Planning Problem

First we define needed variables like ‘Products’, some temporary IntVar variables etc. Then in the resourceAllocation method, we made some calculations and according to these calculations, some primitive constraints are imposed. Each Product variable implemented from zero to (All Materials)/(Needed Materials). Finally we use searching method to all solutions and calculate maximum income for each solution and put it in an ArrayList. At the end we find index of maximum element in income ArrayList, it means that ‘solution: (index+1)’ is correct result. *(Important methods are resourceAllocation and searchMaxIncome methods which are located in BinPackProduct.java file)*

Calculations:

P1	P2	P3	Products
0..4	0..2	0..4	Product Domains
X	y	z	Quantity of Products
x≤ 4	y≤ 4	z≤ 4	Restrictions from total domains

- x*P1 = x*R1 + x*R3
- y*P2 = y*R2 + y*2*R3
- z*P3 = z*R3

As a result:

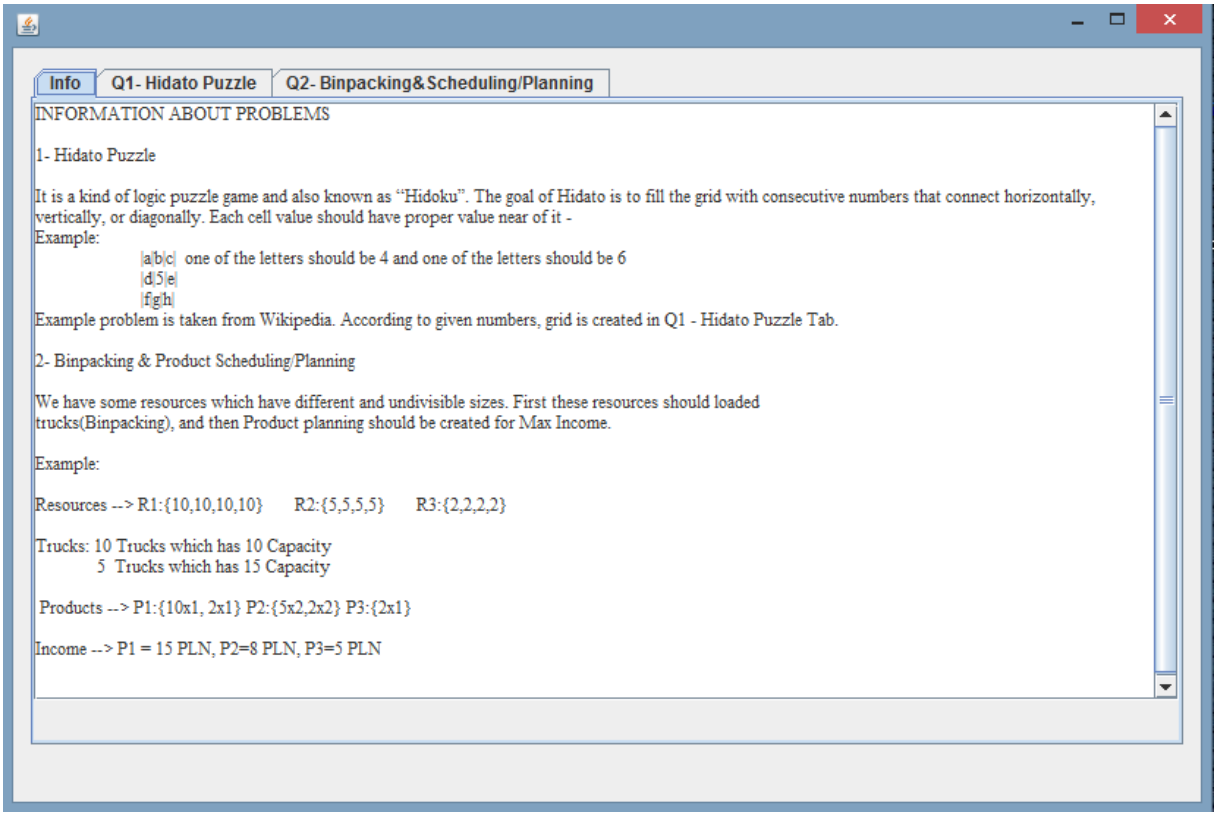
- x ≤ 4 (x*R1 --> max. 4*R1) //this situation handled at definition of domain
- y ≤ 4 (y*R2 --> max. 4*R2) //this situation handled at definition of domain
- x+2y+z ≤ 4 (x*R3+2*y*R3+z*R3) //constraint will imposed for this situation.

Total quantity of P3 can be (0 to 4) because R3 resource is limited.

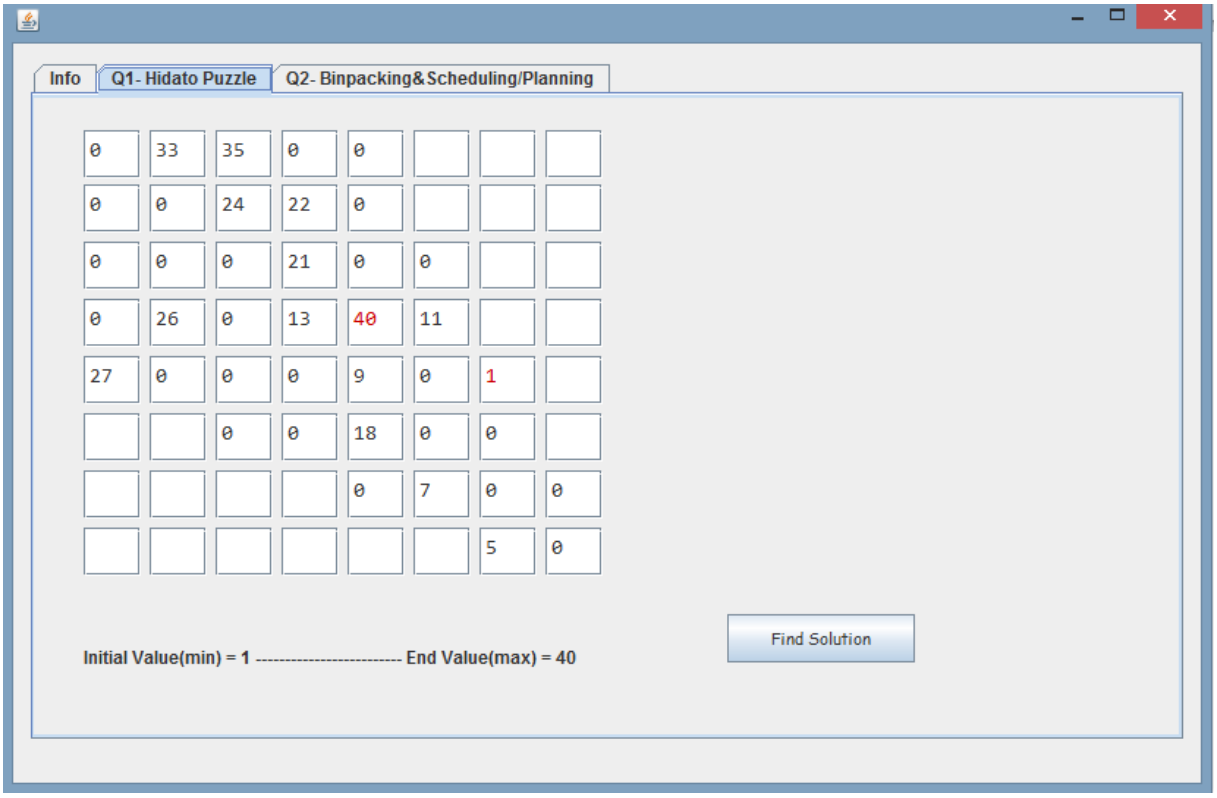
Graphical User Interface

In Graphical interface, there are 3 tab menus.

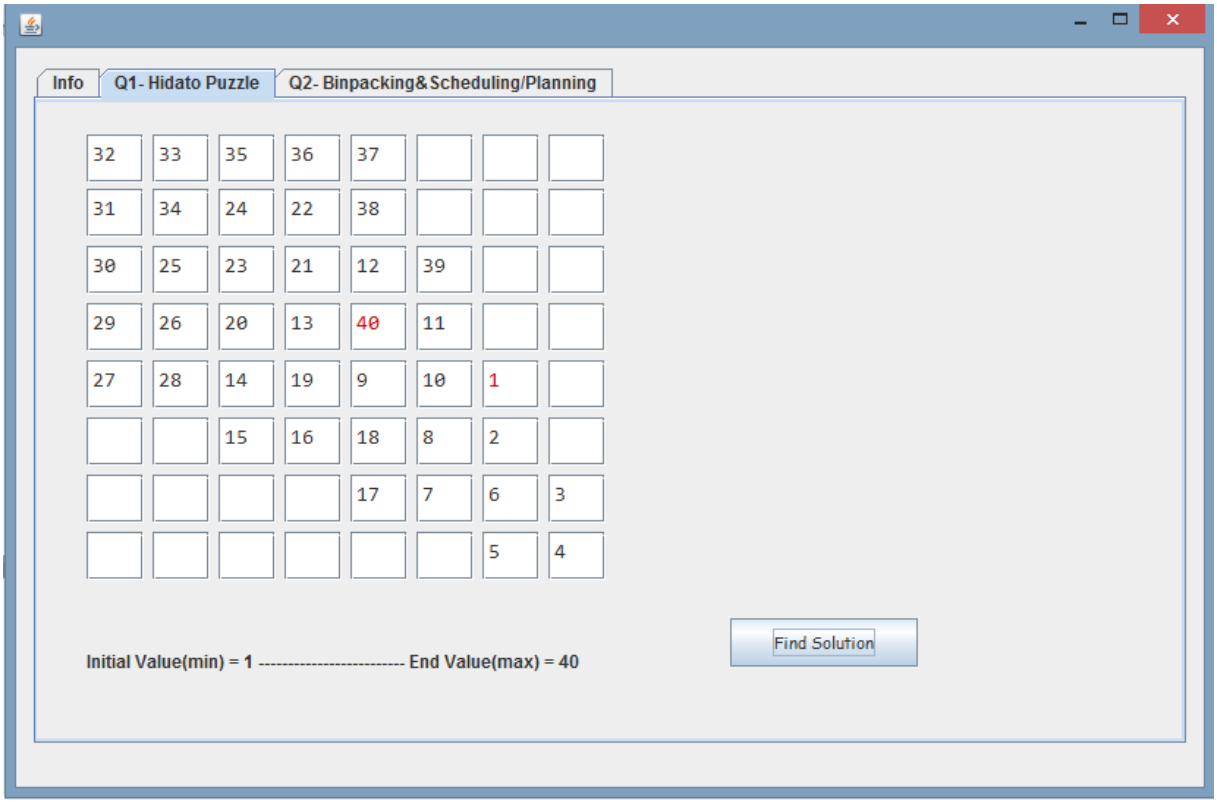
- Fist tab created in order to inform to user about problems. It gives basic information about Hidato puzzle and BinPacking & Scheduling/Planning.



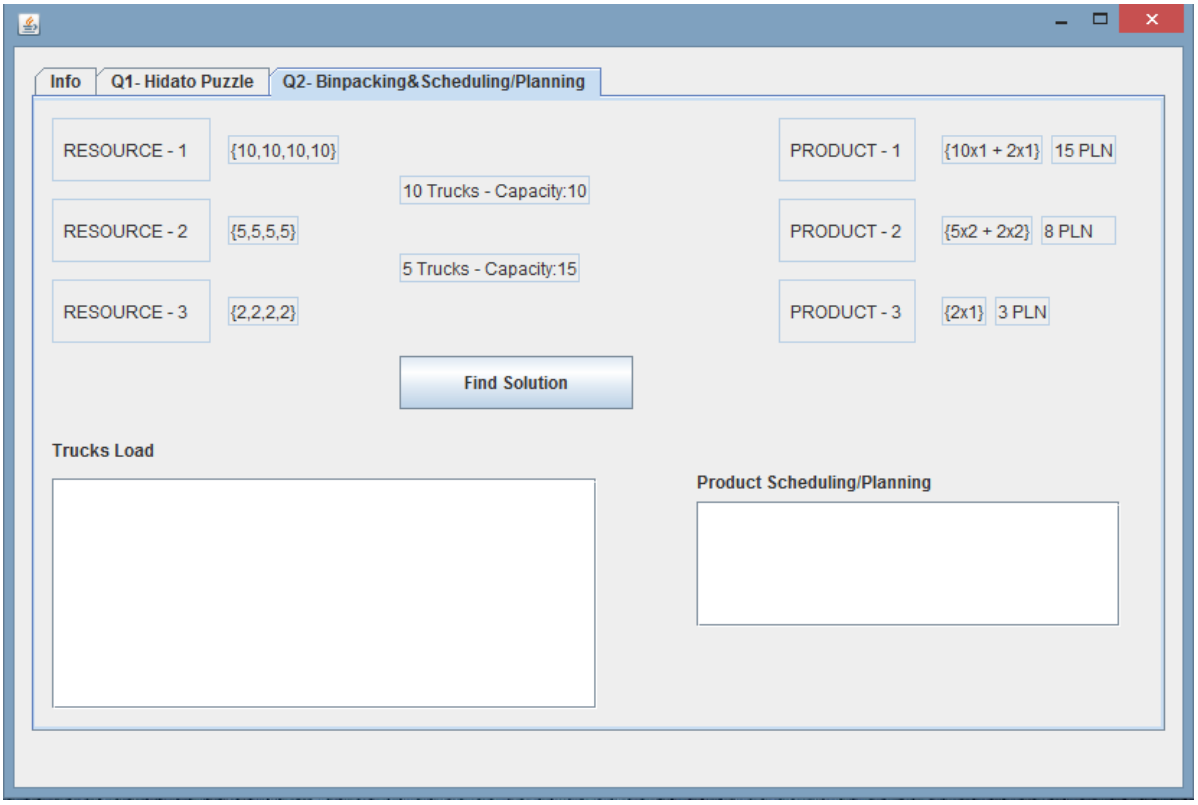
- Second tab is created for Hidato puzzle solution. First situation, it shows initial state of grid and red numbers show minimum and maximum numbers.



After clicking the find solution method, it shows correct solution on grid.



- Third tab is created for Binpacking and Scheduling/Planning solutions. In this tap user can see quantities of resources, products, and trucks.



After clicking 'Find solution' button, user can see results of truck loads and product planning solution for maximum income.